

Open Source

How it works

Developers publish their software online.
System only works when people do more than publish.
They need to contribute.

Definition
Open Source Software is software whose source code is made available under a license that permits users to study, change, and improve the software and to copy and distribute the modified version.

Open Licenses
• GPL
• MIT
• LGPL
• BSD
• Creative Commons

Creates a community

Network Effects

Open Source and Network Effects
Open Source developer ecosystem will not grow without network effects.
Network effects create a virtuous cycle.
Network effects create a virtuous cycle.

Community size
• More contributors → more features → more users → more contributors.
• Network effects create a virtuous cycle.
• Network effects create a virtuous cycle.

Go to contribute
To build community, must create general code software.
Generalized software creates more users.
Doesn't require as well.
Super power.

Network Effects

Two main types for Network Effects:
• Direct Network Effects: The value of the product increases as the number of users increases.
• Indirect Network Effects: The value of the product increases as the number of users increases, but the effect is indirect.

Types

Direct Network Effects
• Network effects that are direct and immediate.
• Network effects that are direct and immediate.

Indirect Network Effects
• Network effects that are indirect and delayed.
• Network effects that are indirect and delayed.

Explanatory Power

• Format wars
• Subsidies to one side of a market
TV - relationship of advertisers and viewers
• Fanboy behavior
Carrying people over
price platform is a rational behavior - more of previous arguments we need

Network Effects are Everywhere

All large categories of industrial and consumer leverage network effects.
Google, Facebook, Amazon, etc.
All parties increase your value.
Your source advantage.
Competition opens up value to win a large win.
Applies to anyone who publishes a platform, where other developers or users create value.

Embedded

Dedicated function

• Router
• TV
• Digital camera
• Set-top box
• Robots

Tension between generalization and specialization

History

Evolution to general purpose OS
• 1960s
• 1970s
• 1980s
• 1990s
• 2000s
• 2010s

General Purpose Hardware

Transition to developing software hardware requires
• Software development
• Hardware development
• Software development
• Hardware development
• Software development
• Hardware development

Internet of Things

IOT Changes the equation

We wear computers in our:
• cars
• appliances
• furniture
• light switches
• clothing
Possibly in our bodies and our food!
In our infrastructure - monitoring environment, water, energy, traffic
Want to run Linux on a 10-cent processor, that runs for years on a single charge

Linux in IOT

Why Linux?
• It's free and open source
• It's secure
• It's reliable
• It's scalable
• It's flexible
• It's easy to use
• It's easy to integrate
• It's easy to maintain
• It's easy to upgrade
• It's easy to customize
• It's easy to deploy
• It's easy to support
• It's easy to learn
• It's easy to teach
• It's easy to share
• It's easy to collaborate
• It's easy to communicate
• It's easy to connect
• It's easy to create
• It's easy to innovate
• It's easy to improve
• It's easy to evolve
• It's easy to grow
• It's easy to thrive
• It's easy to succeed
• It's easy to fail
• It's easy to learn
• It's easy to teach
• It's easy to share
• It's easy to collaborate
• It's easy to communicate
• It's easy to connect
• It's easy to create
• It's easy to innovate
• It's easy to improve
• It's easy to evolve
• It's easy to grow
• It's easy to thrive
• It's easy to succeed
• It's easy to fail

Fragmentation

Bad Fragmentation

Fragmentation leads to inefficiency and waste.
Fragmentation leads to inefficiency and waste.

Good Fragmentation

Fragmentation leads to innovation and growth.
Fragmentation leads to innovation and growth.

Linus Quote

Linus Torvalds: "I don't know if it's a good idea to have a single standard for everything. I think it's better to have a lot of different standards, and let the market decide which one wins."

© 2010-2011 Linus Torvalds
This document is licensed under the Creative Commons Attribution-ShareAlike license.
All rights reserved. No part of this document may be reproduced without prior written permission from Linus Torvalds.

The Paradox of Embedded and Open Source

Tim Bird

Senior Software Engineer, Sony Mobile Communications, Inc
Chair, CE Workgroup Architecture Group
(Founder of ELC)

Keynote

- I haven't keynoted this event for several years
- I don't really have an action for you...



A FILM BY CHRISTOPHER NOLAN

INCEPTION

4/29/2014

FROM THE DIRECTOR OF THE DARK KNIGHT

The Paradox of Open Source and Embedded

ELCeption

- In the movie Inception, an international team implants ideas into their subjects minds, by invading their dreams.
- I want to inject an idea into your mind
 - But, I don't care if you know I'm trying to do it
 - And I don't have my dream-invading equipment with me...

Outline

- Open Source and network effects
- Embedded
- Internet of Things
- Fragmentation

Open Source

How it works

Developers publish their software to the public
 Users only modify when people make changes public
 They need to contribute

Definition

Open source software is software whose source code is made available to the public
 Open source hardware is hardware whose design is made available to the public
 Open source data is data whose source is made available to the public

Creates a community

Network Effects

Collective phenomena within software
 More developers = more users
 More users = more developers
 More developers = more users

Community size

Not a single community
 Top of web community
 (pay not RPS)

Cost to contribute

To build community, must create generalized software
 Generalized software costs more to create
 Elephants perform as well as mice

Open Source and Network Effects

More developers = more users
 More users = more developers
 More developers = more users

Embedded

Dedicated function

- Router
- TV
- Digital camera
- Set-top box
- Robots

Tension between generalization and specialization

History

From custom to general-purpose OS

- Unix
- Windows
- Linux
- Mac OS
- Android
- iOS
- Chrome OS

General Purpose Hardware

Used to build custom hardware

Re-Specialization

Custom hardware for specific functions

Creates a community

They need to contribute

100



size

(if clear-cut)

0-continuous

more discharges = more value

Examples:

- recycled products
- books, clothing
- tools, paint

0-continuous

more


There are multiple sources simultaneously

To build consistency, start creating generalized software

 Generalized software
EASIER TO WRITE

Network Effects

Exponential returns from size effects because the value of participating increases based on the network of participants and users (positive feedback loop).



Types

- Platform
- Product
- Operating System
- Personal Work

Network Effects are Everywhere

All large companies understand and try to leverage network effects.

Google, Facebook, Apple, Microsoft

you provide leverage your value.

First mover advantage

Companies spend billions to win a format war.


Applies to anyone who publishes a platform, where other developers or users create value.

Explanatory Power

- Format wars
- Subsidies to one side of a market
- Fanboy behavior

TV - relationship of advertisers and viewers

Connecting people to one another is a natural behavior - even if emotional arguments are used.



Two-sided markets

Two-sided markets are markets that connect two distinct groups of users.

Examples: eBay, Craigslist, Airbnb, Uber, Lyft, etc.

The diagram illustrates the growth of a network. It starts with a single node on the left, followed by a small cluster of three nodes, then a more complex structure of six nodes, and finally a large, dense, interconnected network of many nodes on the right.

- Format wars
- Subsidies to one side of a market
- TV - relationship advertisers and
- Fanboy behavior

Convincing people to use your platform is a rational choice - even if emotional

```

graph LR
    IOT((Internet of Things)) --- Frag((Fragmentation))
    IOT --- Linux((Linux in IOT))
    IOT --- Eqn((IOT Changes the equation))

    Frag --- Bad((Bad Fragmentation))
    Frag --- Good((Good Fragmentation))

    Linux --- Perf((Performance Linux in IOT))
    Linux --- Stream((Streamlining Linux))

    Eqn --- WeWant((We want computers in our: cars, appliances, furniture, light switches, clothing))
    Eqn --- Possible((Possibly in our bodies and our food!))
    Eqn --- Infra((In our infrastructure - monitoring environment, water, energy, traffic))
    Eqn --- Want10c((Want to run Linux on a 10-cent processor, that runs for years on a single charge))
    Eqn --- Finally((Finally - Linux on a cereal box))
  
```

Internet of Things

Fragmentation

Bad Fragmentation

Good Fragmentation

Linux in IOT

Performance Linux in IOT

- Real-time
- Low power
- Small footprint
- Low cost

Streamlining Linux

IOT Changes the equation

We want computers in our:

- cars
- appliances
- furniture
- light switches
- clothing

Possibly in our bodies and our food!

In our infrastructure - monitoring environment, water, energy, traffic

Want to run Linux on a 10-cent processor, that runs for years on a single charge

Finally - Linux on a cereal box

Possibly in our bodies and our food!

In our infrastructure - monitoring environment, waste, energy, traffic

Want to run lights or process



Finally - Linux on a cer...




Figure 1.10: A screenshot of a presentation slide. On the left is a photo of a man in a patterned shirt. On the right is a list of bullet points in Chinese, discussing the importance of a good first impression and the role of a professional image in business.

A circular flow diagram showing the interaction between firms and households. Firms are represented by blue circles, and households by green circles. Arrows indicate the flow of capital and labor between them.

Stream

Multi-robot Systems

- Top-Down
- Bottom-Up

Multi-robot Engineering

- Multi-robot coordination
- Multi-robot control
- Multi-robot learning
- Multi-robot optimization
- Multi-robot scheduling
- Multi-robot task allocation
- Multi-robot teaming
- Multi-robot training
- Multi-robot validation
- Multi-robot verification



Open Source

How it works



Definition
Open Source Software can be freely used, changed and shared by anyone

Legal framework
GPL License
Guarantees freedom for users and developers to modify and improve the code

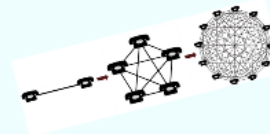
Other Licenses
• BSD
• Apache
• MIT

Different Legal Terms

Creates a community



Network Effects



Open Source and Network Effects

Other developers write software you use

More developers = more value

Ecosystem



Related services:
Books, training,
tools, jobs



Community size



- Not a single community
- Lots of sub-communities

Definition

Open Source Software can be freely used, changed and shared by anyone

GPL License

guarantees freedom
for downstream users
to view source code



framework

Other Licenses

GPL License



Guarantees freedom
for downstream users
to obtain source code

Legal framework

Other Licenses

- BSD
- Apache
- MIT

Different Legal Terms



Open Source

How it works



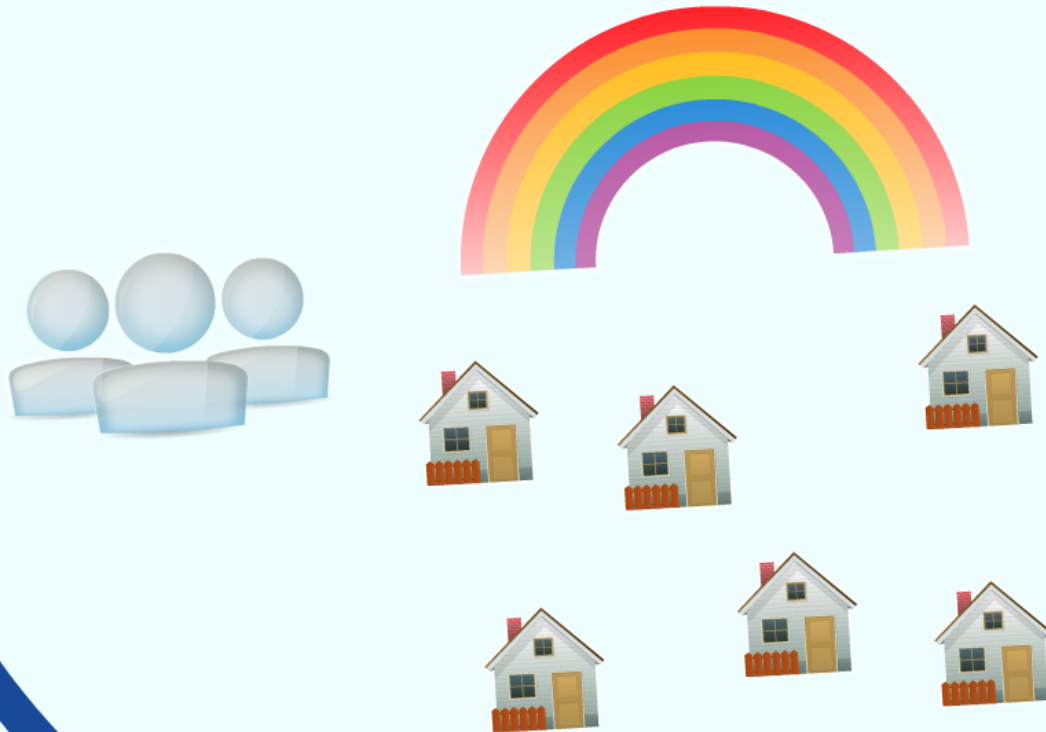
Developers publish their derivative software

System only works when people do more than publish

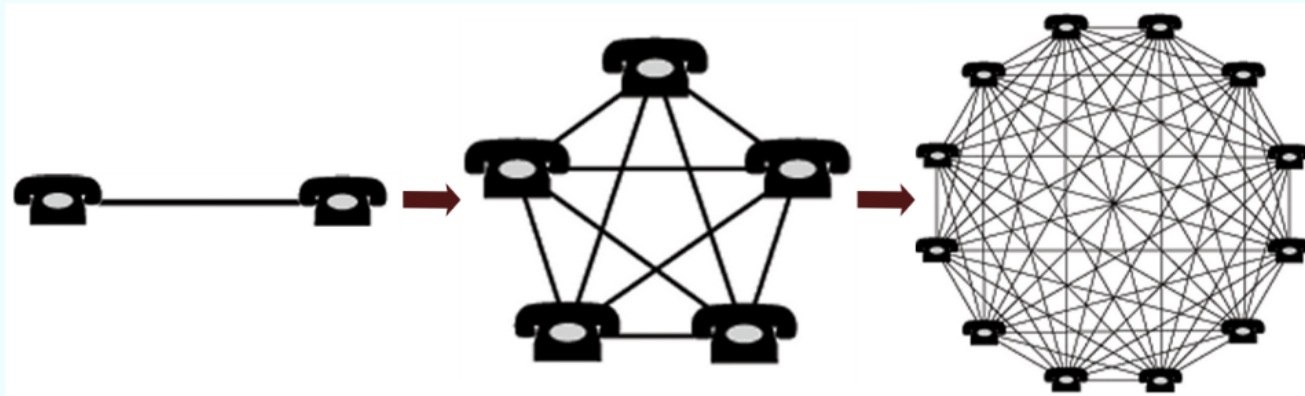
They need to contribute



Creates a community



Network Effects



Network Effects

Economic term for the effect where the value of something increases based on the network of participants (users/developers) that adopt it.



Explanatory Power

- Format wars
- Subsidies to one side of a market

TV - relationship of advertisers and viewers

- Fanboy behavior

Convincing people to use your platform is a rational behavior -- even if irrational arguments are used



Network Effects are Everywhere

All large companies understand and try to leverage network effects

Google



3rd parties increase your value!

First mover advantage

Companies spend billions to win a format war



Applies to anyone who publishes a platform, where other developers or users create value.

Types



Format Wars



Operating Systems



iOS

Two-sided markets



In this -- all involving new creation after initial initial creation

Network Effects



Open Source and Network Effects

Other developers write software you use

More developers = more value



Related services: Books, music, tools, etc.

Community size

- Not a single community
- Lots of sub-communities

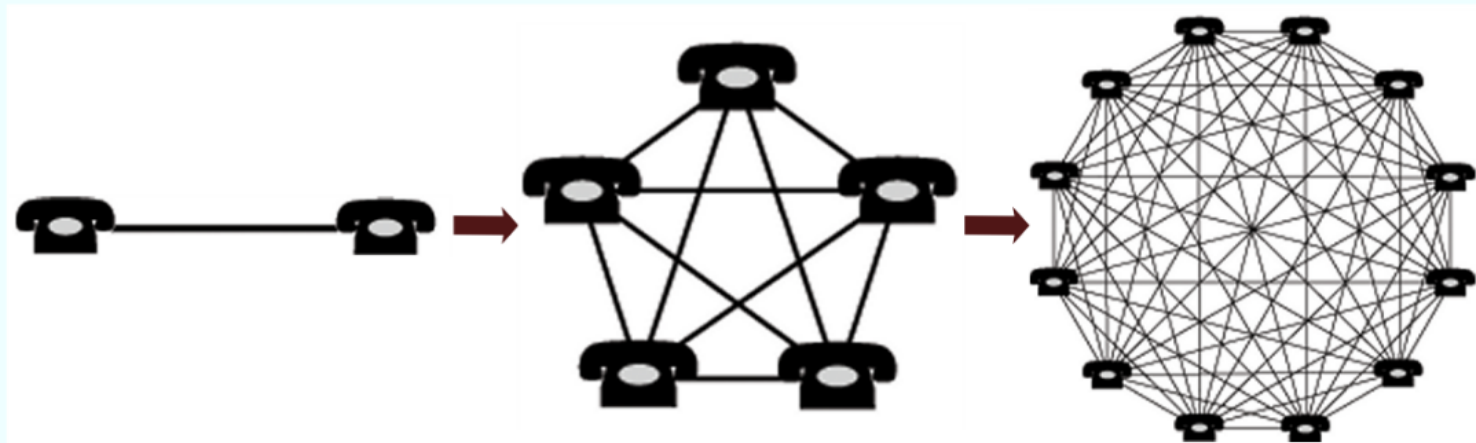
Size matters

Some communities are very small



People are in multiple communities simultaneously

Economic term for the effect where the value of something increases based on the network of participants (users/developers) that adopt it.



Types

Phone network



Format Wars

- VHS vs. Betamax
- HD DVD vs Blu-ray



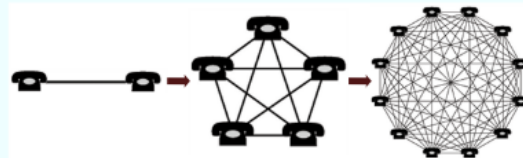
Operating Systems

- Windows vs. Mac vs. Linux
- Android vs. iOS



iOS

Phone network



Operating Systems

- Windows vs. Mac vs. Linux
- Android vs. IOS



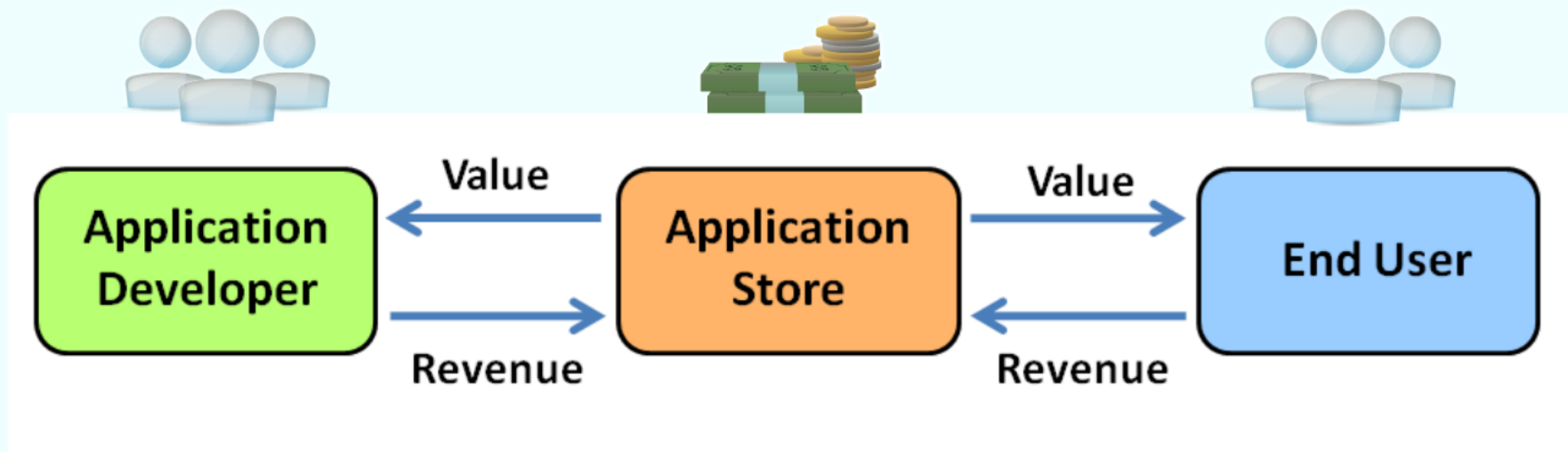
iOS

Format Wars

- VHS vs. Betamax
- HD DVD vs Blu-ray



Two-sided markets



In 2014 -- still evolving our theories
about multi-sided markets

Network Effects are Everywhere

All large companies understand and try to leverage network effects

The Google logo, featuring the word "Google" in its characteristic multi-colored font.The Microsoft logo, featuring the word "Microsoft" in a bold, black, sans-serif font.

3rd parties increase your value!

First mover advantage

leverage network effects

Google



Microsoft

3rd parties increase your value!

First mover advantage

Companies spend billions
to win a format war



3rd parties increase your value!

First mover advantage

Companies spend billions
to win a format war



Applies to anyone who publishes a
platform, where other developers or
users create value.

Explanatory Power

- Format wars
- Subsidies to one side of a market



TV - relationship of
advertisers and viewers

- Fanboy behavior

Convincing people to use
your platform is a rational
behavior -- even if irrational
arguments are used



where

Open Source and Network Effects

Other developers write software you use

More developers = more value

Ecosystem



Related services:
Books, training,
tools, jobs



Community size

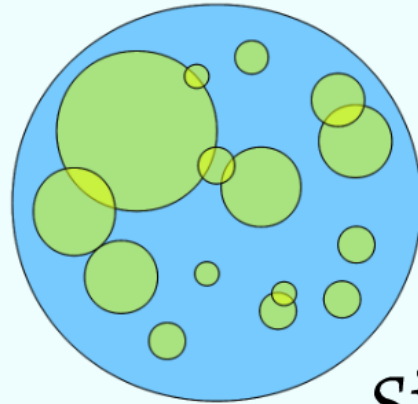
of a single community

Ecosystem



Related services:
Books, training,
tools, jobs

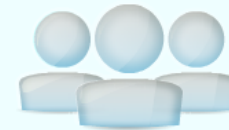
Community size



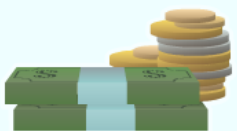
- Not a single community
- Lots of sub-communities

Size matters

Some communities
are very small



People are in multiple
communities simultaneously

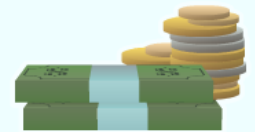


Cost to contribute

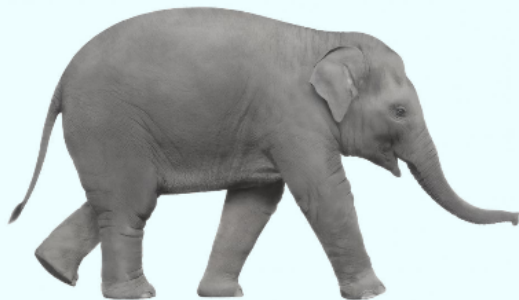
To build community, must create generalized software



Generalized software costs more to write



Doesn't perform as well



Bigger
Slower



Embedded

Dedicated function

- Router
- TV
- Digital camera
- Set-top box
- Robots

Mobile phones



*Tension between generalization
and specialization*



VS



Re-Specialization




Andrew Murray's talk on Bo
(2010 ELC Europe)



Amulet

History

From custom to general-purpose OS

- VxWorks
 - Nucleus
 - pSOS
 - VRTX
 - LynxOS
 - ITRON
 - QNX
- 
- Linux

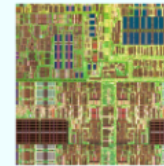


Cheaper, more ca
hardware

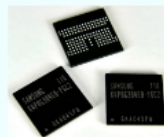
General Purpose Hardware

Tradeoff in development time vs. hardware resources

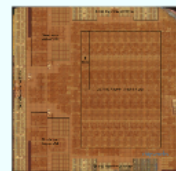
Modern SOC's have enormous complexity



Cheapest DRAM is 32M



CPU with 9 cores is same cost as one with 3 cores



Get used to wasting silicon!

, more capable
re

- Ex.
- Work. Exports
- 4 years of work
- Good track record
- Tight timeline after the
- I also want to share my observations

Re-Specialization



Andrew Murray's talk on Boot Time
(2010 ELC Europe)

The Right Approach to Minimal Boot Times

Andrew Murray
Senior Software Engineer, NPC Data

- Driver and kernel development
- Embedded applications development
- Windows driver development

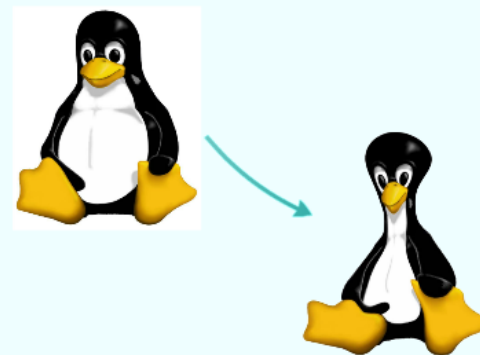
Work Experience

- 4 years of experience working with embedded U
- Good track record in dramatically reducing costs through NPC Data's best time reduction service
- Tight timescales often doesn't permit nice, fit solutions - however this frustration had provided ideas I wish to share today
- I also wish to share my observations and my time reduction

The swiftBOOT Approach

- Further improvements can be gained in different times:
- Parallelisation
- Using Argon's async framework
- Deferred loading of less important
- Loadable kernel modules

Linux Kernel (Before and After)



These Hardware
vs. hardware resources

Device Tree



Not a rant
OK - a little bit of a rant

DT Problems

- No typing
- Separation of data from code
 - Harder to write drivers
- Language problems
- Unfamiliarity
- Multi-node dependencies not easy-to-express

DT is Hard to specialize

It's meant to support single image

Kernel parses tree at runtime

Can't do compile-time optimizations

Relate here my long sad story about Link-Time Optimisation and how parsed data items are not optimizable by the compiler.

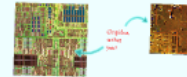
DT and network effects

DT helps build network effects



Has encouraged restructuring platform code for re-use

Exposes IP blocks between platforms



all code
drivers

not

DT is Hard to specialize

It's meant to support
single image

Kernel parses tree
at runtime

Can't do compile-
time optimizations

*Relate here my long sad story about Link-Time
Optimization and how parsed data items are not
optimizable by the compiler.*

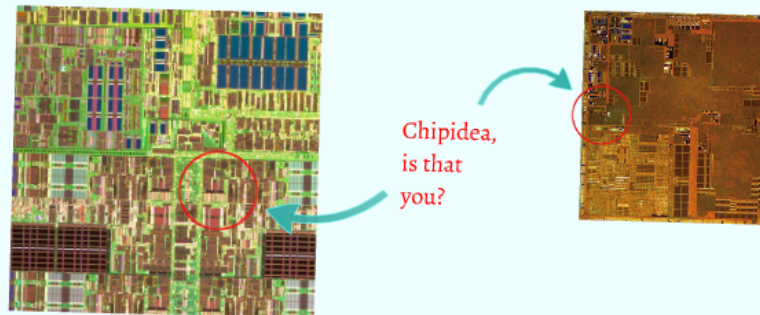
DT and network effects

DT helps build network effects

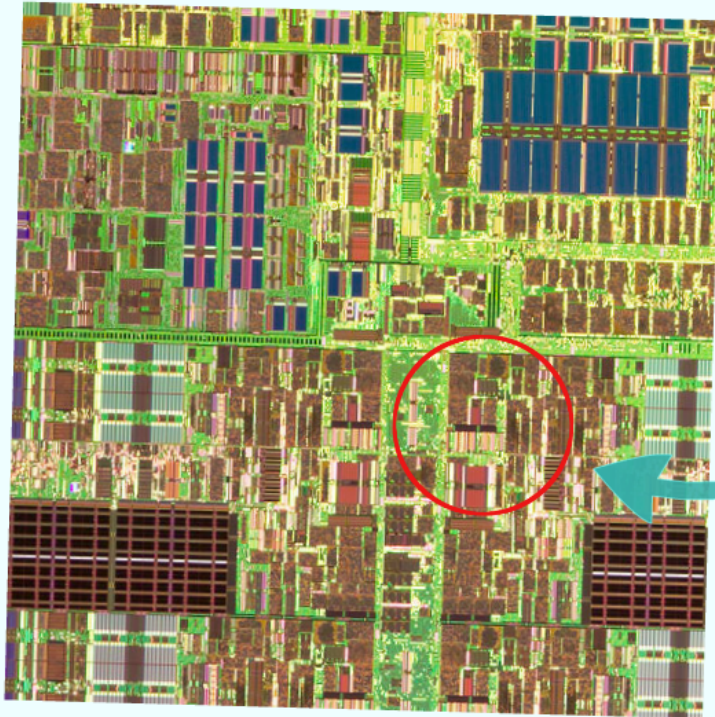


Has encouraged restructuring
platform code for re-use

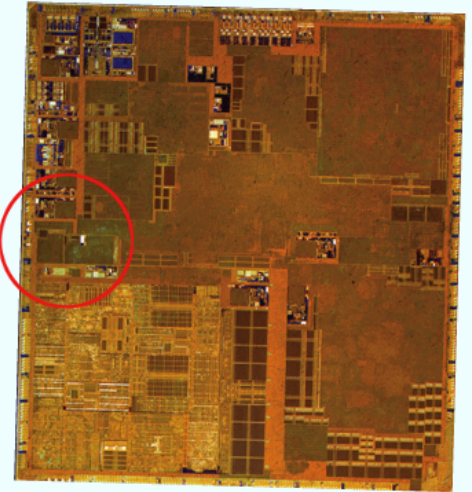
Exposes IP blocks between platforms



Exposes IP blocks between platforms



Chipidea,
is that
you?



Internet of Things

IOT Changes the equation

We want computers in our:

- cars
- appliances
- furniture
- light switches
- clothing



Possibly in our bodies
and our food!



In our infrastructure - monitoring
environment, water, energy, traffic



Want to run Linux on a 10-cent
processor, that runs for years
on a single charge

Linux in IOT

Why Linux?

Do we actually need Linux here?



Reuse

What reuse are we talking for?

- Embedded Linux
- Embedded Linux
- Embedded Linux
- Embedded Linux



Streamlining Linux

Modularity/Legos

Large blocks of software



Subtractive Engineering

Infrastructure engineering



Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Infrastructure engineering

Finally - Linux on a cereal box



Linus Quote



Linux Mail - Ask Linus, May 2000

IOT

Changes the equation

We want computers in our:

- cars
- appliances
- furniture
- light switches
- clothing



Possibly in our bodies
and our food!



In our infrastructure - monitoring
environment, water, energy, traffic



Want to run Linux on a 10-cent
processor, that runs for years
on a single charge

Pro

Linux is too
• Big
• Slow
• Power-hungry
• Insecure

Finally - Linux on a ceral box



Linux in IOT

Why Linux?

Do we actually need Linux here?



Reuse

What re-use are we striving for?

- People want to leverage:
- Network stack
 - File systems
 - USB
 - NFC

SOC support



Streamlining Linux

Modularity/Legos

Lego Model of Software



Subtractive Engineering

Folly of subtractive engineering



As system scales up, it's harder to remove than build from scratch

Nobody wants to remove stuff they don't understand



Just say "NO" to subtractive engineering

Problem with Linux in IOT

Linux is too:

- Big
- Slow
- Power-hungry
- Insecure

Linux 0.11 system
ran in 2MB

More features since then

Lose community

If we slim down Linux,
it's not Linux anymore

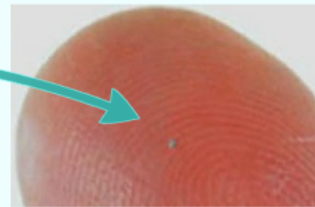
No network effects

in our bodies
Food!



Why Linux?

Do we actually need Linux here?



Hitachi rfid chip

Problem with Linux in IOT

Linux is too:

- Big
- Slow
- Power-hungry
- Insecure

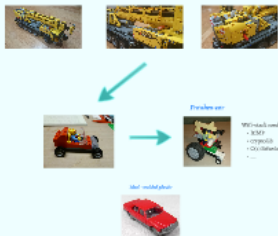
Linux 0.11 system
ran in 2MB

More features since then

Streamlining Linux

Modularity/Legos

Lego Model of Software



Subtractive Engineering

Folly of subtractive engineering



As system scales up, it's harder to remove than build from scratch

Nobody wants to remove stuff they don't understand



Just say "NO" to subtractive engineering

Lose community

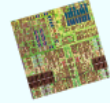
If we slim down Linux, it's not Linux anymore

No network effects

What is...
People want to leverage...

- Network stack
- File systems
- USB
- NFC

SOC support



Modularity/Legos

Lego Model of Software



Franken-car



Wifi-stack needs:

- ICMP
- crypto lib
- O(1) Scheduler
-

Ideal - molded plastic

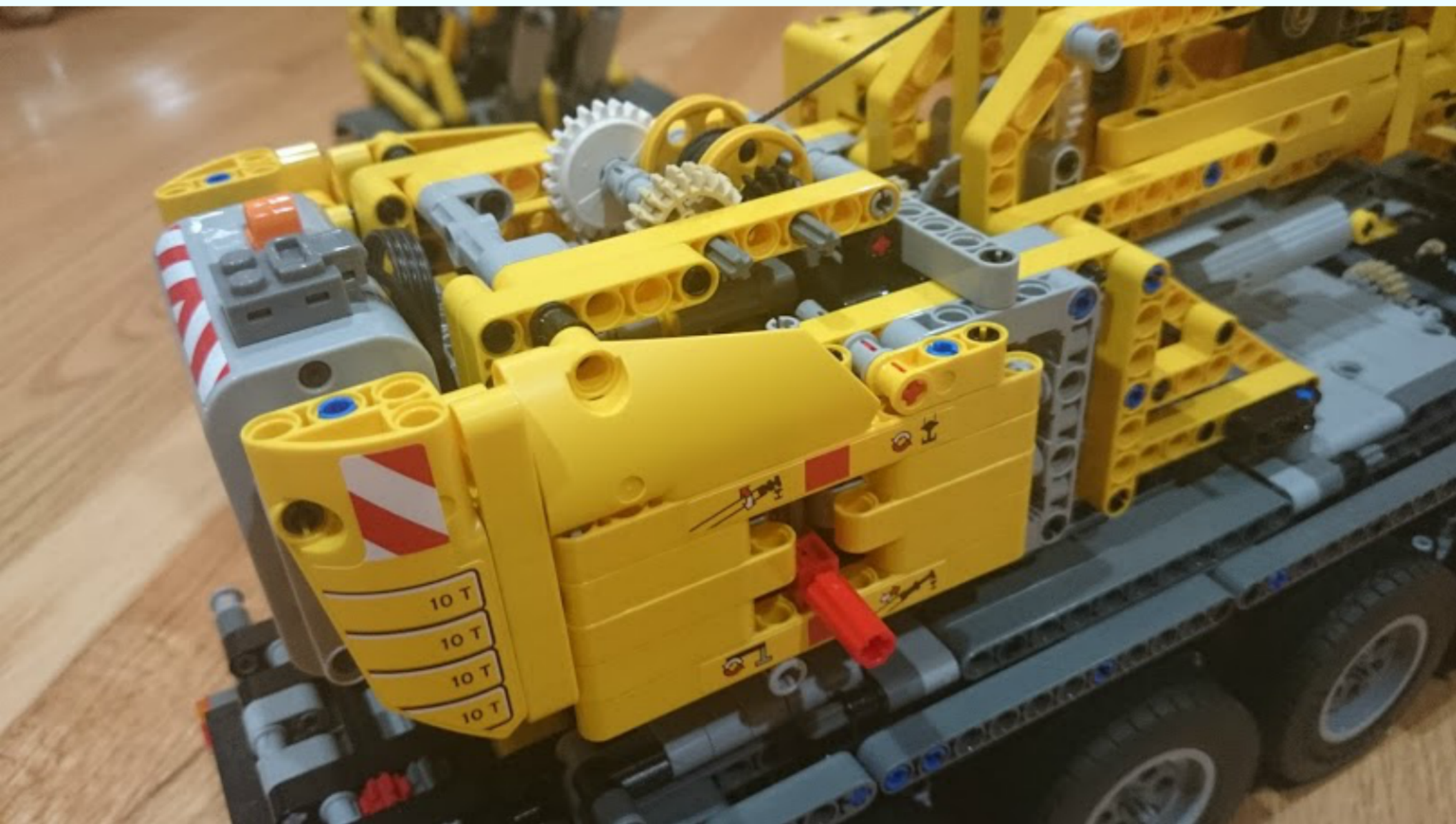


Su

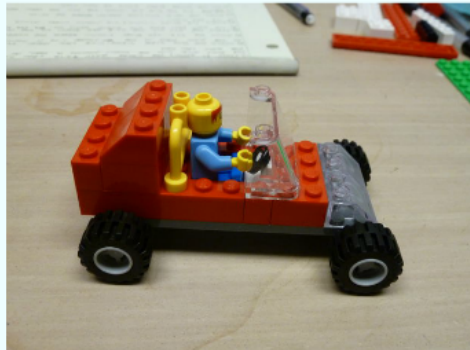
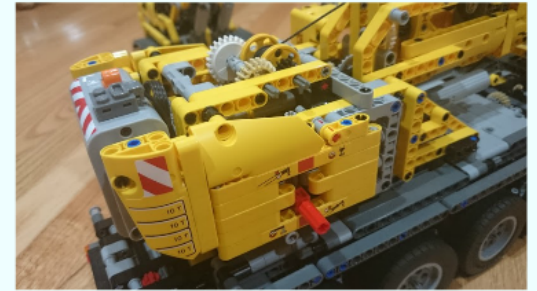
Folly of sub







Lego Model of Software

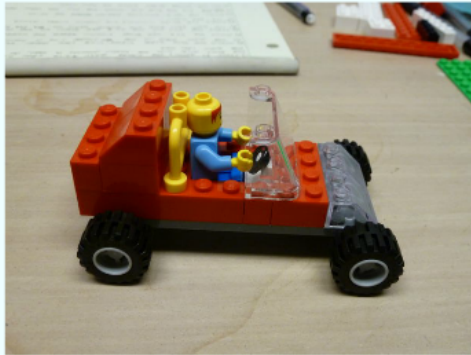


Franken-car



Wifi-stack needs:

- ICMP
- crypto lib
- $O(1)$ Scheduler
-



Franken-car



- Wifi-stack needs:
- ICMP
 - crypto lib
 - O(1) Scheduler
 -

Ideal - molded plastic





Franken-car



Wifi-stack needs:

- ICMP
- crypto lib
- O(1) Scheduler
- ...

ed plastic



Subtractive Engineering

Folly of subtractive engineering



As system scales up, it's harder to remove than build from scratch

Nobody wants to remove stuff they don't understand



Just say "NO" to subtractive engineering

If we slim d
it's not Lin

No netwo



Lose community

If we slim down Linux,
it's not Linux anymore

No network effects

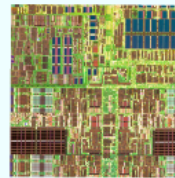
Reuse

What re-use are we striving for?

People want to leverage:

- Network stack
- File systems
- USB
- NFC

SOC support



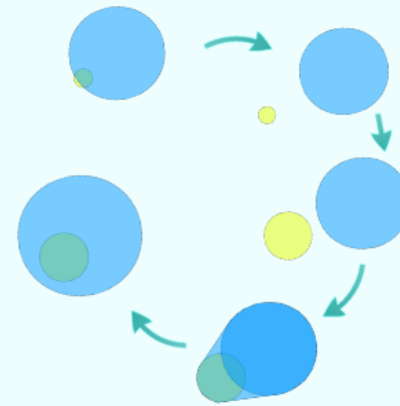
What to do?



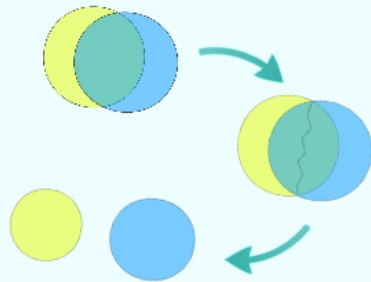
Fork!

Fragmentation

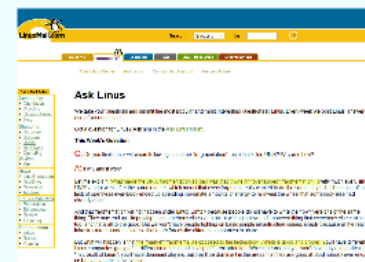
Good Fragmentation



Bad Fragmentation



Linus Quote

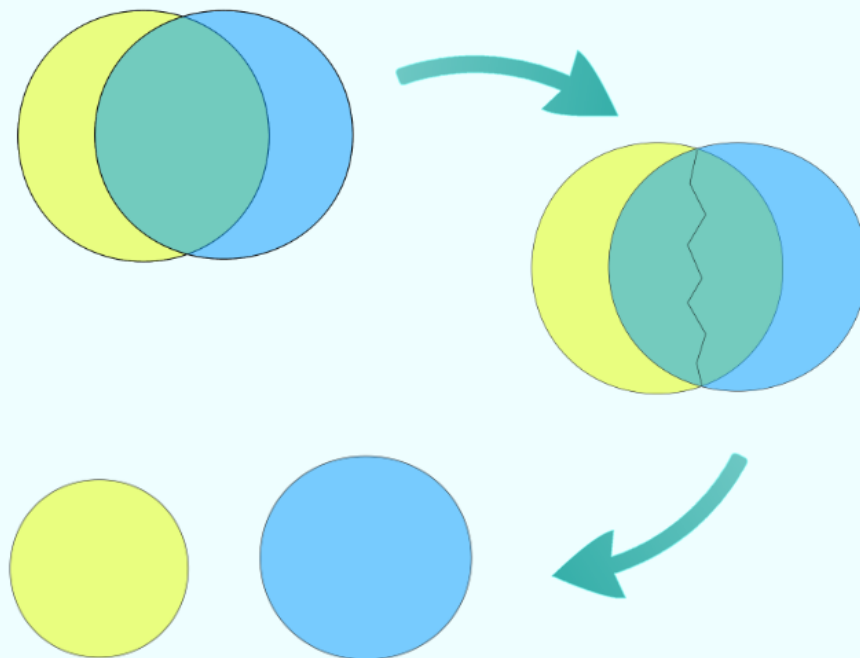


What to do?

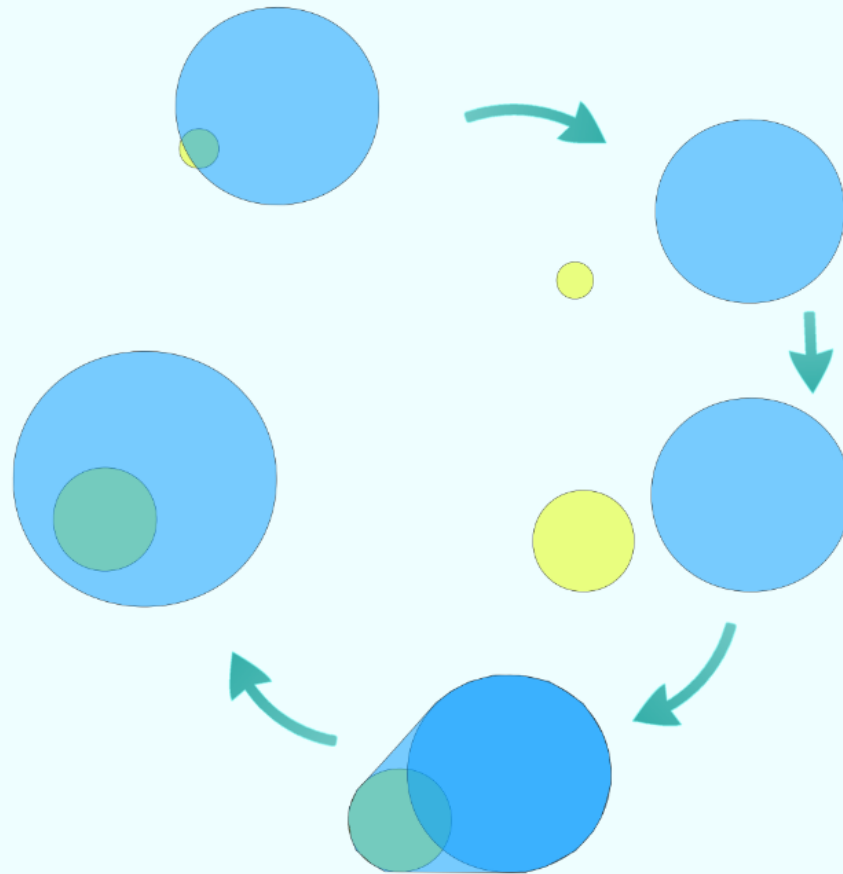


Fork!

Bad Fragmentation




Good Fragmentation



Linus Quote





[Webstore](#)
[Community](#)
[Shopping](#)
[Books](#)
[Linux Resources](#)
[Solutions/Answers](#)

[Ordering Guide](#)
[Ask Linux](#)
[Community Directory](#)
[FAQs/Power](#)

QuickIndex

- Community
- Discussion
- Directory
- Linux Power
- Forum
- Shopping
- Hardware
- Software
- Users
- Help Desk
- Questions
- System
- Fun
- News
- LinuxMall.com
- Linuxies
- Hardware
- Audio/Video
- Linux Resources
- File Services
- Education
- Books
- Linux.org
- Entertainment
- Linux
- Games

Ask Linus

We take your questions and submit the most popular and most interesting questions to Linus. Every week we post Linus' answer to one of your questions.

Got a question for Linus? Ask him in the [Ask Linux](#) forum

This Week's Question

Q: Do you think Linux will avoid following the same "fragmentation" route that killed UNIX? Why and how?


A: Well, and it won't

Let me explain. What made the UNIX fragmentation so bad was that it was an "overlapped" fragmentation: pretty much every single UNIX vendor would enter the same market, which meant that every fragment really wanted to do the same things, but because of the lack of openness every body ended up spending inordinate amounts of energy to re-invent the wheel that somebody else had already done.

And that fragmentation will not happen under Linux. Simply because people do not have to write their own versions of the same thing. They may end up "improving" on somebody else's version, or writing a new and improved thing that everybody else can use too, and that's all to the good. But we won't have people fighting an basic simple infrastructure issues, simply because all the real infrastructure is under a license that really forces the different companies to share.

But what will happen is that the "market" fragments, as opposed to the technology. Which is good and proper. You'll have different companies going after different markets, and trying different practices. Which means that you won't have any one particular "microsoft of Linux" you'll have dominant players, but they'll be dominant in the areas that they are good at. And nobody ever ends up being the bad at everything.

This week's LinuxMall.com's **Ask Linux** column is sponsored by Team Linux.



Hot Items
[View Cart](#)

[VShop by UltraLAI](#)

[Practical K&C](#)

[\\$23.99](#)

Linux Mall - Ask Linus, May 2000

We take your questions and submit the most popular and most interesting questions to Linus. Every week we post Linus' answer to one of your questions.

Got a question for Linus? Ask him in the [Ask Linus forum](#).

This Week's Question

Q: Do you think Linux will avoid following the same "fragmentation" route that killed UNIX? Why and how?

A: It will, and it won't.

Let me explain. What made the UNIX fragmentation so bad was that it was an "overlapped" fragmentation - pretty much every single UNIX vendor went after the same market, which meant that every fragment really wanted to do the same things, but because of the lack of openness everybody ended up spending inordinate amounts of energy to re-invent the wheel that somebody else had already done.

And that fragmentation will not happen under Linux. Simply because people do not have to write their own versions of the same thing. They may end up improving on somebody else's version, or writing a new and improved thing that everybody else can use too, and that's all to the good. But we won't have people fighting on basic simple infrastructure issues, simply because all the real infrastructure is under a license that really forces the different companies to share.

But what will happen is that the "market" fragments, as opposed to the technology. Which is good and proper. You'll have different Linux companies going after different markets, and having different priorities. Which means that you won't have any one particular "microsoft of Linux": you'll have dominant players, but they'll be dominant in the areas that they are good at. And nobody ever ends up being the best at everything.

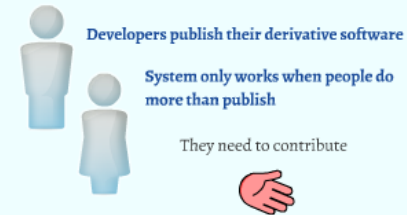
This week's LinuxMall.com's **Ask Linus** column is sponsored by Team Linux.



Linux Mall - Ask Linus, May 2000

Open Source

How it works



Definition

Open Source Software can be freely used, changed and shared by anyone

Legal framework

Copyright, freedom of information, patents, etc.

Other Licenses

- BSD
- Apache
- MIT

Different Legal Terms

Creates a community



Network Effects



Embedded

ated function

camera
box

Mobile phones

between generalization
lization

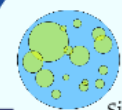
Re-Specialization



Cost to contribute

To build community, must create

Community size



- Not a single community
- Lots of sub-communities

Size matters

Some communities are very small

People are in multiple communities simultaneously

Open Source Network

Other developers write you use

More dev

Ecosystem

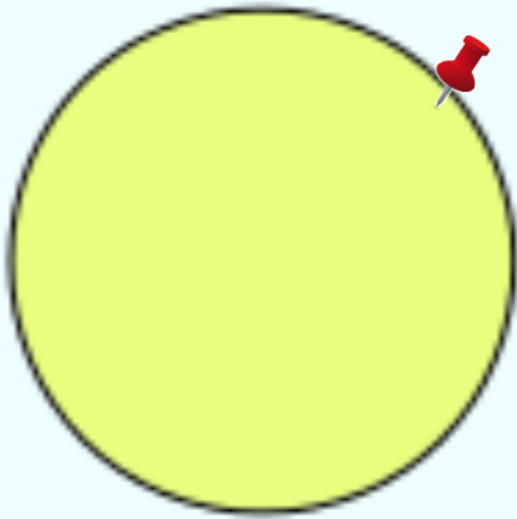


Related services
Books, training
tools, jobs

At the intersection of Open Source and Embedded



**We need a new base camp
here!**



ELCeption

- Pay attention to network effects
- Forking can equal growth

Thanks for your time